(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 November 2001 (15.11.2001)

PCT

(10) International Publication Number
WO 01/86389 A2

(51) International Patent Classification[7]: G06F 1/00

(21) International Application Number: PCT/US01/13374

(22) International Filing Date: 26 April 2001 (26.04.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/568,771 10 May 2000 (10.05.2000) US

(71) Applicant: MOTOROLA, INC. [US/US]; 1303 East Algonquin Road, Schaumburg, IL 60196 (US).

(72) Inventors: PERONA, Richard, Allen; 1502 W. Islandia Drive, Gilbert, AZ 85233 (US). WILLIAMS, Clifford, Andrew; 22807 N. 30th Avenue, Phoenix, AZ 85027 (US).

(74) Agents: INGRASSIA, Vincent, B.__ Motorola, Inc. et al.; P.O. Box 10219, Scottsdale, AZ 85271-0219 (US).

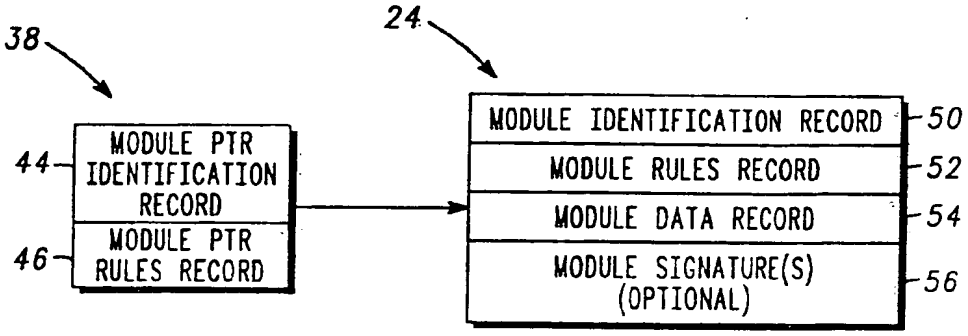
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SOFTWARE-DEFINED COMMUNICATIONS SYSTEM EXECUTION CONTROL



(57) Abstract: Software execution control in which a series of two-way rule checks is performed between software-defined communications system component records to ensure and maintain system security and integrity. A system platform (20) performs a series of two-way rule checks between records of a system platform (20) and an application (22) called by the platform (20), between records of the called application (22) and a module (24) that defines the called application (22), and between the records of the module (24) that defines the called application (22) and the platform (20). Both the called application (22) and the module (24) that defines the called application (22) are then instantiated if the two-way rule checks are successful. Because the rule checks are performed in a two-way manner, restrictions such as licensing and source restrictions may be placed not only on system modules (24-30), but also on the applications (22) using the modules (24-30), thereby enabling higher levels of system security to be achieved. In addition, the present invention minimizes processing overhead by providing for load-time rule checking rather than run-time checking associated with conventional enforcement systems.

WO 01/86389 A2

originated from other vendors. In addition, the techniques typically perform checking during execution of the modules or application, and are therefore not capable of asserting additional rules prior to execution to increase system integrity.

Brief Description of the Drawings

Additional objects and advantages of the present invention will be more readily apparent from the following detailed description of preferred embodiments thereof when taken together with the accompanying drawings in which:

FIG. 1 is a block diagram illustrating a first exemplary open architecture system including the execution control of the present invention;

FIG. 2 is a block diagram of the software components of the execution control of the present invention;

FIG. 3 is a block diagram showing a module pointer record and module of FIG. 2 in greater detail;

FIG. 4 is a block diagram illustrating the sequence of rule checks among records of system components when a system platform calls a system application; and

FIG. 5 is a block diagram illustrating a second exemplary open architecture system including the execution control of the present invention.

Detailed Description of the Preferred Embodiments

Referring now to the drawings in which like numerals reference like parts, FIG. 1 shows components of a software-defined open architecture communications system 10 of the type in which the present invention is implemented. The system 10 has numerous hardware and software components that can be individually removed, replaced, upgraded and/or otherwise modified without having to correspondingly modify all other system components. According to a preferred embodiment of the present invention, the system 10 is a Wireless Information Transmitting System (WITS) in the form of a radio designed and sold by Motorola Corp., the assignee of the present invention. Such a system may interface to a wide variety of other communications devices such as, for example, internet portals

such as a personal computer 12, wireless communications devices such as a cellular phone 14, and a communications satellite 16, as well as other WITS systems (not shown).

The operation of each of the components in the above-described radio 10 is defined by software that is pre-loaded into the radio and then typically upgraded on a periodic basis. The software itself is composed of numerous components that may be bundled together and provided by a single vendor, or, more typically, individually provided by two or more vendors. In the latter situation, the execution control of the present invention enables software components from multiple vendors to be individually loaded, upgraded or replaced in a manner that ensures that only components that are licensed or otherwise approved for use with one another may be utilized in combination. Such an open architecture system provides system designers with a high degree of flexibility both in maintaining the system and in modifying the system as system communications requirements change, while at the same time maintaining the underlying integrity of the radio 10.

FIG. 2 shows exemplary software components of the radio 10 of FIG. 1. The components include a computing platform 20, an application 22 and several modules 24, 26, 28, 30. However, it should be appreciated that the number of applications and modules may vary depending on the specific underlying computer platform. Each of these software components 20-30 interacts with others of the components to define the operation of the radio 10 pursuant to the execution control of the present invention as will be described below in more detail.

The computing platform 20 is hardware based and is the operating system of the radio 10. In the above-discussed WITS radio system, the computing platform 20 includes platform identification information that uniquely identifies the platform when checked by other software components as will be discussed later, and rules information that includes conventional types of rules such as required application endorsements, module endorsements and capacity constraints, as well as other vendor-specific rules concerning the application 22 or modules 24-30 such as, for example, locality of use or period of use rules.

In FIG. 2, the application 22 defines an algorithm that enables the radio 10 to execute a pre-defined function. The application 22 is defined by a series of records,

may be validated by the platform 20 at the time of loading the module 24 onto the platform 20.

With reference now to FIG. 4, operation of the present invention will now be described with respect to the communications device 10, the computing platform 20, the application 22 and the module 24. Specifically, the series of two-way rule checks executed by the execution control of the present invention among system components during loading of the application 22 and the module 24, and therefore prior to application run-time, will now be described. In the following discussion of the operation of the present invention, the term "rule check" is used to refer to the validation of numerous rules and other requirements that must be met by some or all of the system components during application/module loading and prior to application run-time. Such rules/requirements may include source authentication, certification/endorsement status, platform capabilities, record corruption status, and security clearance status rules and requirements. However, the exact requirements imposed by a system component on other components may vary according to system and vendor needs.

Initially, at 60, the platform 20 receives a user request to load and execute the application 22 and subsequently checks the application identification record 32 against the platform rules and configuration information and parses the module pointer record 38. Additionally, the platform 20 may also check the application signature record 48 if an extra measure of security/integrity is desired. At the same time, at 62 the platform 20 checks the integrity of the platform rules and configuration information to determine per the application rules record 34 if the platform 20 is authorized through, for example, a vendor license agreement, to load the application 22.

Upon receiving the loading command from the platform 20, at 64 the platform checks the contents of the module identification record 50 against the module pointer rules record 46 to verify both the integrity and the source of the module 24. Also, the platform 20 at 66 accesses and checks the integrity of the module 24 by checking the module identification record 50. Subsequently, the platform 20 at 68 checks the integrity of both the application 22 per the application identification record 32 and itself at 70 per the platform rules and configuration information

against the module rules record 52 to determine if both the application 22 and the platform 20 meet all requirements of the module 24.

If each of the above two-way rule checks is successful, the platform 20 completes instantiation of the application 22 and the module 24, and execution of the underlying application can then be carried out. If, however, any of the rule checks performed at 60-70 fails, the platform terminates loading the application 22. In other words, if, for example, the application 22 determines via the rule check at 64 that the module identification record file 50 does not contain the necessary licensing agreement between the application vendor and the module vendor, or that the module security or data has been compromised, the application 22 will not allow the platform 20 to download the module 24, and instantiation of the application 22 will be terminated.

At this point it should be understood that, while FIG. 4 shows a series of two-way rule checks among the platform 20, the application 22 and the module 24, the checks at 62, 64, 68 and 70 and as described above are actually performed by the platform 20. More specifically, the platform 20 must load the application 22 and the module 24 in conjunction with the rules contained in the platform configuration and rules information, the application rules record 34, the module pointer rules record 46 and the module rules record 52. Therefore, the application 22 and the module 24 must trust the platform 20 to perform the checks at 62, 64, 68 and 70, to terminate loading of the application 22 if any of the checks fail, and to remove any part of the application 22 that has been loaded if loading of the application 22 is terminated.

In addition, the checks performed by the platform at 60 and 66 may include the validation of digital signatures stored in the application and module signature records 48, 56, respectively.

While only the methodology of the present invention has been described with reference to the exemplary platform 20, application 22 and module 24, it should be appreciated that the execution control of the present invention may be utilized in an open software environment including any number of applications and modules. It should also be appreciated that, in a multiple application system, certain of the modules may be utilized by more than one application. FIG. 5 shows an example of a communications device 100 including a platform 120, two separate applications